CLAIMS

What is Claimed is:

1. A method comprising:

stalling execution of a function, said function having an associated first state of a stack frame;

saving a copy of said first state of said stack frame;

initiating execution of said function;

stalling completion of said function, said function having an associated second state of said stack frame;

comparing said second state of said stack frame to said copy of said first state of said stack frame; and

determining whether said stack frame is corrupted based on said comparing.

2. The method of Claim 1, further comprising:

upon a determination that said stack frame is corrupted, replacing said second state of said stack frame with said copy of said first state of said stack frame; and

initiating completion of said function.

3. The method of Claim 2, wherein said stalling execution of a function comprises:

stalling completion of a prologue of said function.

4. The method of Claim 3, wherein said initiating execution of said function comprises:

initiating completion of said prologue.

5. The method of Claim 3, wherein said prologue is a standard prologue.

6. The method of Claim 2, wherein said stalling completion of said function comprises:

stalling completion of an epilogue of said function.

7. The method of Claim 6, wherein said initiating completion of said function comprises:

initiating completion of said epilogue.

8. The method of Claim 6, wherein said epilogue is a standard epilogue.

9. The method of Claim 2, further comprising:

generating a notification of a stack frame corruption.

10. The method of Claim 1, further comprising:

upon a determination that said stack frame is not corrupted, initiating completion of said function.

11. A method comprising:

stalling completion of a prologue of a function, said function having an associated first state of a stack frame;

saving a copy of said first state of said stack frame to a location in a memory;

initiating completion of said prologue, wherein execution of said function is permitted;

stalling completion of an epilogue of said function, said function having an associated second state of said stack frame;

locating said copy of said first state of said stack frame;

comparing said second state of said stack frame to said copy of said first state of said stack frame;

determining whether said stack frame is corrupted based on said comparing;

upon a determination that said stack frame is corrupted, replacing said second state of said stack frame with said copy of said first state of said stack frame; and

initiating completion of said epilogue, wherein completion of said function is permitted.

12. The method of Claim 11, wherein said prologue is a standard prologue and said epilogue is a standard epilogue.

13. The method of Claim 11, wherein said determining whether said second state of said stack frame is corrupted comprises:

determining whether said second state of said stack frame is the same as said first state of said stack frame; and

upon a determination that said second state of said stack frame is not the same as said first state of said stack frame, determining that said stack frame is corrupted.

14. The method of Claim 13, wherein said determining whether said second state of said stack frame is the same as said first state of said stack frame comprises:

performing a byte to byte comparison of bytes in said second state of said stack frame to bytes in said first state of said stack frame;

upon a determination that said bytes of said second state of said stack frame match said bytes of said first state of said stack frame, said second state of said stack frame is determined to be the same as said first state of said stack frame; and

upon a determination that said bytes of said second state of said stack frame do not match said bytes of said first state of said stack frame, said second state of said stack frame is determined not to be the same as said second state of said stack frame.

15. A method comprising:

hooking a standard prologue of a function, said function having an associated stack frame in a stack;

hooking a standard epilogue of said function;

stalling completion of said standard prologue, wherein execution of said function is stalled, said function having an associated first state of said stack frame;

saving a copy of said first state of said stack frame to a location in memory different from said stack;

initiating completion of said prologue, wherein execution of said function is initiated;

stalling completion of said epilogue, wherein completion of said function is stalled, said function having an associated second state of said stack frame;

locating said copy of said first state of said stack frame;

comparing said second state of said stack frame to said copy of said first state of said stack frame;

determining whether said stack frame is corrupted based on said comparing; and

upon a determination that said stack frame is corrupted, replacing said second state of said stack frame with said copy of said first state of said stack frame; and

initiating completion of said epilogue, wherein completion of said function is permitted.

16. The method of Claim 15, the method further comprising:

generating a notification of a stack frame corruption.

17. The method of Claim 15, wherein said prologue is a standard prologue and said epilogue is a standard epilogue.

18. The method of Claim 15, wherein said location in said memory different from said stack is a heap.

19. The method of Claim 15, further comprising:

saving an address of said location in said memory different from said stack to a hash table using a value of an EBP associated with said function in said stack as a pointer to said address.

20. The method of Claim 19, wherein said copy of said first state of said stack frame is located using said EBP as a pointer to said address in said hash table.

21. A computer-program product comprising a computer-readable medium containing computer program code comprising:

a stack frame corruption detection and recovery application for determining whether a stack frame corruption of a stack frame associated with a function has occurred based upon a comparison of a first state of said stack frame with a second state of said stack frame; and

upon a determination that said stack frame corruption has occurred, said stack frame corruption detection and recovery application for replacing said second state of said stack frame with said first state of said stack frame to permit continued availability of said function beyond said stack frame corruption.